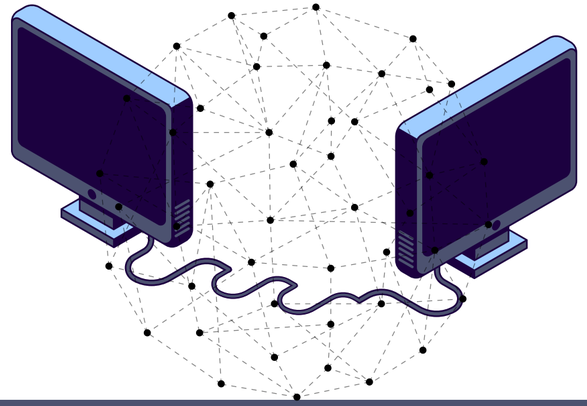
ENHANCED LEAKY BUCKET CONGESTION CONTROL IN TCP TRANSMISSION

TABLE OF CONTENTS

INTRODUCTION TO CONGESTION

What is Congestion?

Congestion occurs when the network nodes or links carry more data than their capacity, leading to packet loss and delays.



Impact:

- Increased latency
- Packet drops
- Reduced throughput

Example:

Sudden surge in traffic during events like online sales or streaming a live concert.

BENEFITS OF CONGESTION CONTROL

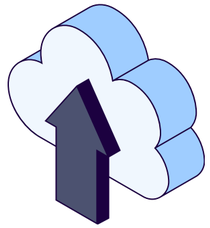
Ensures Optimal Resource Usage:
Prevents overloading network links and balances traffic flow.

Improves Network Reliability:
Maintains stable connections by avoiding packet collisions and retransmissions.

Reduces Delays and Packet Drops:
Controls traffic to minimize queuing delays and data loss in transit.

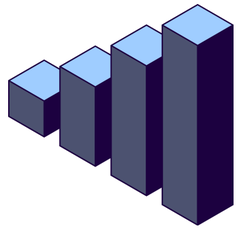
CONSEQUENCES OF UNCONTROLLED CONGESTION

Packet loss



Data packets are dropped when the network is overloaded, leading to incomplete transmission.

Decreased efficiency



Excessive retransmissions waste network resources, slowing down overall performance.

Poor user experience

Increased delays and disruptions result in frustration for users, especially in real-time applications like video calls.

MISCONCEPTIONS ABOUT CONGESTION

Common Misconceptions:

Congestion primarily results from high traffic demand exceeding network capacity, leading to delays and packet loss. While increasing bandwidth can help by providing more capacity, it doesn't fully resolve congestion if other factors like network protocols or routing inefficiencies are also contributing. Thus, bandwidth increase alone may not always be a complete solution.

Reality-

Congestion can arise from inefficient algorithms that fail to manage traffic effectively, causing delays. A balanced approach, combining increased bandwidth with optimized control mechanisms like traffic shaping and congestion control, is necessary to prevent and resolve congestion efficiently.

OBJECTIVES

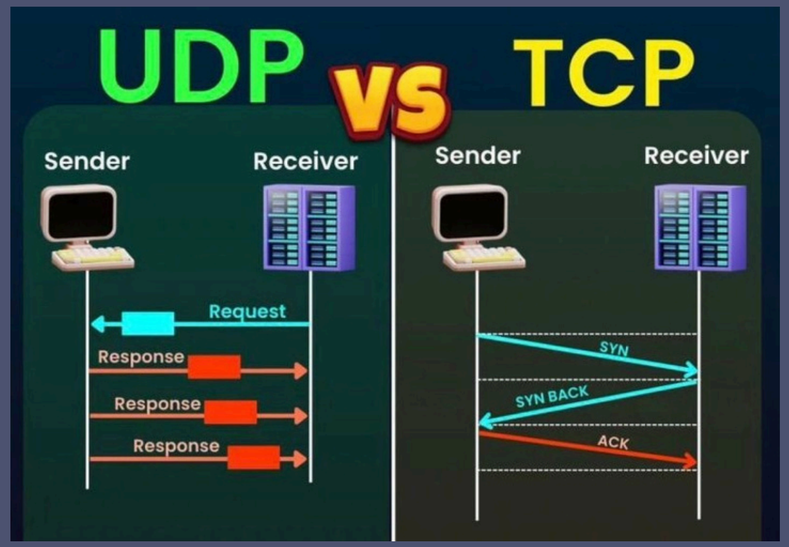
GENERAL OBJECTIVE

The general objective of congestion control in a network is to prevent network overload by managing traffic flow, ensuring optimal resource utilization, and maintaining stable performance.

SPECIFIC OBJECTIVE

The objective is to manage excessive incoming traffic by utilizing a one dynamic bucket methodology, where one buckets is divided into 2 and used to handle and regulate the traffic flow efficiently. This approach ensures optimal distribution and prevents overload in the system.

TCP OVER UDP



Key Differences:

- **TCP (Transmission Control Protocol):** Reliable, connection-oriented, ensures data delivery with error checking, retransmission, and congestion control.
- **UDP (User Datagram Protocol):** Unreliable, connectionless, suitable for real-time applications where speed is prioritized over accuracy, such as streaming or online gaming.

Why TCP?

- Ensures data integrity: TCP guarantees the reliable delivery of data by using acknowledgments and error-checking mechanisms.
- Congestion control mechanisms: It adjusts the rate of data transmission to prevent network congestion and ensure efficient communication.

LEAKY BUCKET ALGORITHM

The Leaky Bucket algorithm is a traffic-shaping mechanism that ensures data transmission at a consistent rate by queuing incoming packets and releasing them steadily. If the bucket (queue) overflows due to excessive incoming traffic, the excess packets are dropped, preventing network congestion.

Analogy-

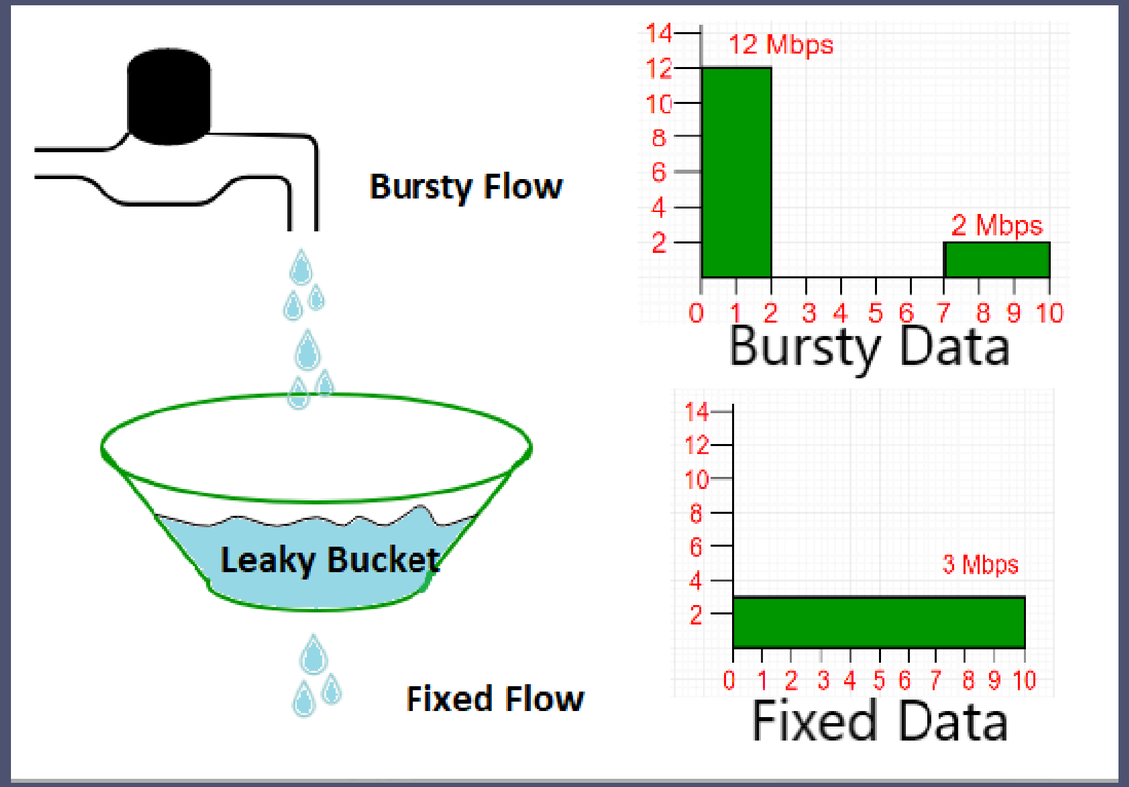
Imagine pouring water into a bucket with a hole at the bottom. The water leaks out at a constant rate, irrespective of how it enters.

HOW THE LEAKY BUCKET WORKS

Steps:

- Incoming packets are stored in the bucket, which leaks data at a constant rate, irrespective of the bursty input.
- If the bucket overflows due to excessive input, extra packets are dropped, maintaining network stability.
- Bursty traffic is converted into a steady stream, ensuring consistent data flow.

- **Example Scenario:** A source sends data at variable rates, but the Leaky Bucket ensures a constant output, e.g., regulating an 3 Mbps flow for 3 seconds instead of sudden bursts.



ADVANTAGES OF THE LEAKY BUCKET ALGORITHM

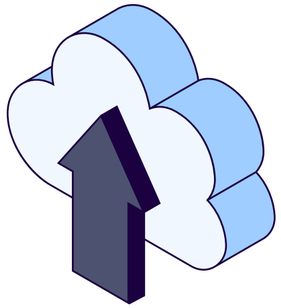
Prevents network overload: Smooths out traffic bursts to maintain a steady flow.

Easy implementation: Simple design makes it easy to deploy and manage.

Reduces jitter: Ensures consistent data transmission for better performance.

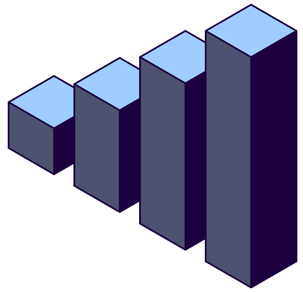
LIMITATIONS OF THE LEAKY BUCKET ALGORITHM

Packet loss



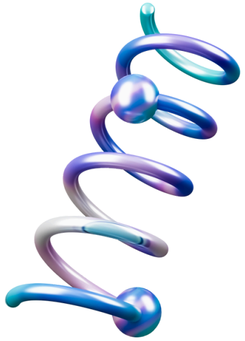
Drops packets when the bucket overflows

Rigidity



Cannot adapt to varying traffic patterns.

Lack of Flexibility



Not suitable for applications requiring dynamic rates

Leaky Bucket vs Token Bucket

Features	Leaky Bucket	Token Bucket
Bucket Capacity	Fixed size bucket that leaks at a constant rate	Bucket capacity can vary, tokens generated at a fixed rate
Packet Transmission Rate	Fixed rate, regardless of traffic bursts	Variable rate, allows bursty traffic up to token limit
Behavior for Burst Traffic	No burst handling; excess traffic is discarded	Allows bursts if there are enough tokens, otherwise packets wait
Token Generation	No tokens; just a fixed leak rate	Tokens generated at a fixed rate
Handling of Excess Traffic	Excess traffic is discarded if the bucket overflows	Excess tokens can accumulate, allowing for burst transmission
Usage	Suitable for smoothing traffic over time	Suitable for handling both bursty and steady traffic

METHODOLOGIES
TWO-BUCKET LEAKY BUCKET APPROACH

Concept:

- Combines two leaky buckets, one regulating incoming traffic and the other shaping outgoing traffic.

Working:

1. Bucket 1: Controls bursty traffic from the sender by smoothing it at a steady rate.
2. Bucket 2: Ensures controlled delivery to the receiver, reducing overflow.
3. This dual approach helps handle higher traffic loads while maintaining consistent throughput.

Example:

- In video conferencing, one bucket regulates the video stream rate, while the second ensures smooth packet delivery for consistent playback.

CHALLENGES FACED

Synchronizing the two buckets effectively.

Handling high traffic loads during testing.

Fine-tuning parameters for different network scenarios.

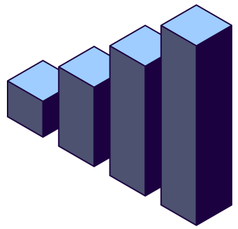
Increase Delay due to high memory usuage

LIMITATIONS OF TWO-BUCKET APPROACH

Complexity

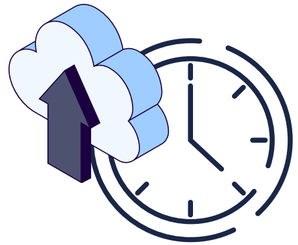
Requires more computational resources for synchronization between the two buckets.

Scalability Issues



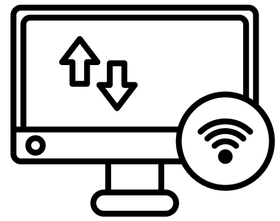
May struggle to handle extremely high traffic volumes in large-scale networks.

Latency



Slight delays can occur due to added regulation layers, especially under heavy loads.

Packet Loss Risk:



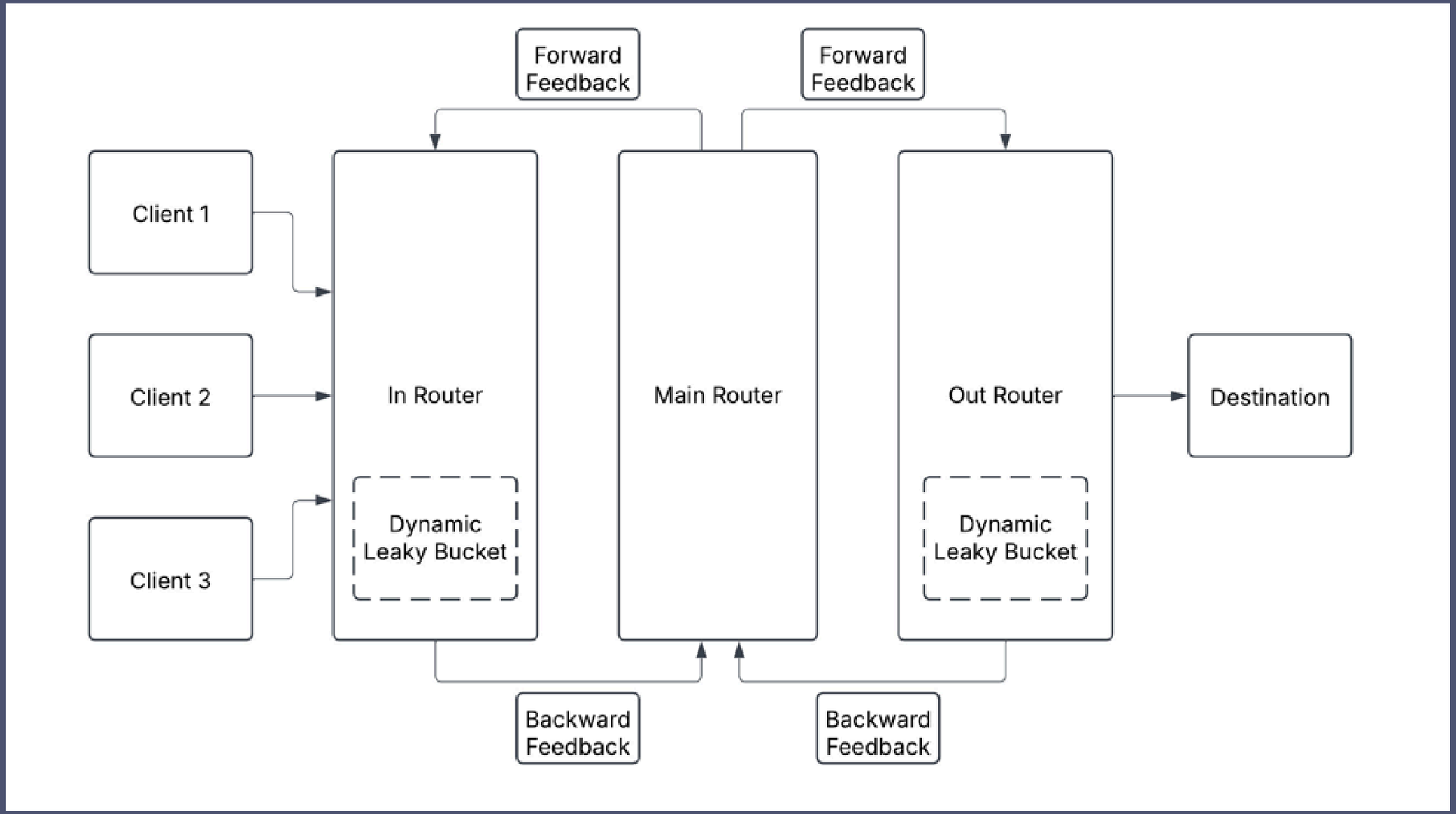
If both buckets overflow simultaneously, packet loss is unavoidable.

CURRENT APPROACH
DYNAMIC ONE BUCKET APPROACH

New Approach and changes to the old methodologies:

- Integrating the Two-Bucket Approach into Dynamic One Bucket for higher efficiency.
- The total bucket is divide into 2 parts: first half is the active Leaky Bucket handling packets, and second half is the reserved bucket space for the first half.
- Thus having a safety net if the original bucket overflows, the reserve storage comes in as a emergency extra space.

ARCHITECTURE DIAGRAM



ARCHITECTURE DIAGRAM

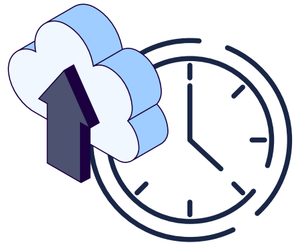
- **Client 1, Client 2, Client 3:** Sources sending data into the system.
- **In Router:** Handles incoming data and regulates flow.
- **Dynamic Leaky Bucket (In Router):** Controls input rate to prevent overload.
- **Main Router:** Core processor routing data efficiently.
- **Out Router:** Prepares and forwards data to the destination.
- **Dynamic Leaky Bucket (Out Router):** Controls output rate for stability.
- **Forward Feedback:** Helps to adjust and optimize data flow based on downstream conditions.
- **Backward Feedback:** Helps to adjust and optimize data flow based on upstream conditions.
- **Destination:** Final endpoint for processed data.

LIMITATIONS OF DYNAMIC ONE BUCKET APPROACH

Space Complexity

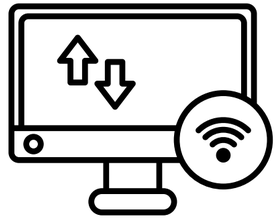
Requires more hardware resources for extra storage

Latency



Slight delays can occur due to added extra storage

Packet Loss Risk:



Limitation of Leacky Bucky is stiil exists. If the incoming packet is larger than the Total Bucket Size then it will be dropped.

THANK YOU